Using Node.js to download files

Posted on October 27th, 2011 under Node.js
Tags: Buffer, CURL, HTTP, node.js, Stream, wget

How to download files using Node.js

There are three approaches to writing a file downloader app using Node - i. HTTP.get, ii. curl, iii. wget. I have created functions for all of them. To get the examples working makes sure you have the dependencies and app variables intact. Read the comments thoroughly, you will not only learn how to download files, but will also learn more about Node's `child_process`, File System, Buffers, and Streams. Let's start with HTTP.get.

Downloading using HTTP.get

HTTP.get is Node's built-in mechanism for making HTTP GET requests, which can also be used for downloading files using the HTTP protocol. The advantage of using HTTP.get is that you don't rely on any external programs to download files.

```
// Dependencies
var fs = require('fs');
var url = require('url');
var http = require('http');
var exec = require('child_process').exec;
var spawn = require('child_process').spawn;

// App variables
var file_url = 'http://upload.wikimedia.org/wikipedia/co
var DOWNLOAD_DIR = './downloads/';

// We will be downloading the files to a directory, so m
// This step is not required if you have manually create
var mkdir = 'mkdir -p ' + DOWNLOAD_DIR;
var child = exec(mkdir, function(err, stdout, stderr) {
    if (err) throw err;
    else download_file_httpget(file_url);
});

// Function to download file using HTTP.get
var download_file_httpget = function(file_url) {
var options = {
    host: url.parse(file_url).host,
    port: 80,
    path: url.parse(file_url).pathname
};
```

```
var file_name = url.parse(file_url).pathname.split('/').
var file = fs.createWriteStream(DOWNLOAD_DIR + file_name

http.get(options, function(res) {
    res.on('data', function(data) {
            file.write(data);
        }).on('end', function() {
            file.end();
            console.log(file_name + ' downloaded to ' +
        });
    });
};
```

The above function is probably the best way to download files using HTTP.get in Node. Make a HTTP.get request and create a writable stream using `fs.createWriteStream`. Since the HTTP.get's response is a stream, it has the `data` event, which carries the chunks of data sent by the server. One each `data` event, write the data to the writeable stream. Once the server finishes sending data, close the instance of `fs.createWriteStream`. If you are trying to use `fs.write` or `fs.writeFile` or any of their variants, they will fail for medium to large files. Use `fs.createWriteStream` instead for reliable results.

Downloading using curl

To download files using `curl` in Node.js we will need to use Node's `child_process` module. We will be calling `curl` using `child_process`'s `spawn` method. We are using `spawn` instead of `exec` for the sake of convenience - `spawn` returns a `stream` with `data` event and doesn't have buffer size issue unlike `exec`. That doesn't mean `exec` is inferior to `spawn`; in fact we will use `exec` to download files using `wget`.

```
// Function to download file using curl
var download_file_curl = function(file_url) {

    // extract the file name
    var file_name = url.parse(file_url).pathname.split('
    // create an instance of writable stream
    var file = fs.createWriteStream(DOWNLOAD_DIR + file_
    // execute curl using child_process' spawn function
    var curl = spawn('curl', [file_url]);
    // add a 'data' event listener for the spawn instanc
    curl.stdout.on('data', function(data) { file.write(d
    // add an 'end' event listener to close the writeabl
    curl.stdout.on('end', function(data) {
        file.end();
        console.log(file_name + ' downloaded to ' + DOWN
    });
    // when the spawn child process exits, check if ther
    curl.on('exit', function(code) {
        if (code != 0) {
            console.log('Failed: ' + code);
        }
    });
};
```

The way data was written to the instance of `fs.createWriteStream` is similar to way we did for HTTP.get. The only difference is that the `data` and `end` events are listened on the `stdout` object of `spawn`. Also we listen to `spawn`'s exit `event` to make note of any errors.

Downloading using wget

Although it says downloading using `wget`, this example applies to downloading using `curl` with the `-O` option too. This method of downloading looks the most simple from coding point of view.

```
// Function to download file using wget
var download_file_wget = function(file_url) {

    // extract the file name
    var file_name = url.parse(file_url).pathname.split('
    // compose the wget command
    var wget = 'wget -P ' + DOWNLOAD_DIR + ' ' + file_ur
    // excute wget using child_process' exec function

    var child = exec(wget, function(err, stdout, stderr)
        if (err) throw err;
        else console.log(file_name + ' downloaded to ' +
    });
};
```

In the above method, we used `child_process`'s `exec` function to run `wget`. Why `exec` and not `spawn`? Because we just want `wget` to tell us if the work was done properly or not, we are not interested in buffers and streams. We are making `wget` do all the dirty work of making request, handling data, and saving the file for us. As you might have guessed, this method is the fastest among the three methods I described.

So now the question is - which method is the best? The answer - whatever suits your need. The `wget` method is probably the best is you want to save the files to the local disk, but certainly not if you want to send those files as a response to a current client request; for something like that you would need to use a stream. All the three methods have multiple options, you choice will ultimately depend on what your needs are. Happy downloading!

Further Reading

1. Node.js HTTP
2. Node.js fs
3. Node.js Child Processes
4. Node.js Buffers
5. Node.js Streams

Related to this post

i. Difference between spawn and exec of Node.js child_process
ii. How to download file using cURL

iii. How to install wget on your Mac
iv. Command line Node.js Programs / Scripts / Utilities / Modules
v. Node.js Restart on File Change
vi. Node.js – Amazon S3: How to get Started
vii. Node.js EventEmitter Tutorial
viii. Install Node.js and NPM on Windows
ix. Sourceforge download does not start – Force it anyway
x. How to install Node.js on Webfaction

10 Responses to “Using Node.js to download files”

Robin Houtevelts says:
December 7, 2011 at 1:15 pm
Thanks,

This was helpful.

---

cielo says:
December 18, 2011 at 12:44 am
Thank you. That was really helpful.

Vladimir says:
January 18, 2012 at 12:55 pm
Thanks a lot!
You are awesome person for sharing this code.

0x80 says:
January 24, 2012 at 7:59 am
Thanks for the code. I had a problem with the http get version because my file url included a port. The host in options will also include the port. Fixed it like this:

```
var options = {
host: url.parse(file_url).host.split(“:”)[0],
port: 80,
path: url.parse(file_url).pathname
};
```

A small note, it’s safer to use the path module for constructing paths. I normally don’t

include a training / in the folder name and using the path.join function you never have to worry about it:

path.join(DOWNLOAD_DIR, file_name)

0x80 says:
January 24, 2012 at 8:24 am
Forgot something. If you get a specific port, don't use 80!

var options = {
host: url.parse(file_url).host.split(":")[0],
port: url.parse(file_url).host.split(":")[1] || 80,
path: url.parse(file_url).pathname
};

Miguel Ribeiro says:
February 13, 2012 at 4:00 am
Hi there.
Is there any way to have an "error" listener?
If i lose connection to internet for a long time while the file is downloading, it won't resume.
Any ideas?
Thanks for you code btw

Captain says:
February 15, 2012 at 12:11 am
Hola Miguel, since we are dealing with streams – http://nodejs.org/docs/latest/api/streams.html – you can add a listener to the 'error' event of the stream.

```
curl.stdout.on('error', function(err) {
console.log(err);
// restart the downloader
});
```

Hope that helps.

---

amit says:
March 2, 2012 at 1:36 pm
this is a very good post…thanks a lot

Arkar says:
June 19, 2012 at 4:54 am
ReferenceError: url is not defined ..
.. on Wget method.

---

cmanzana says:
July 30, 2012 at 1:05 am
Hi all,

one thing that confuses me with spawn in all the code examples that I have seen is that how is it guaranteed that you will get all the events that it triggers:
- we first spawn
- after that we attach the event listeners

what happens if the event is triggered before we attached the listener?

somehow all the examples I have seen on spawn seem to assume that this is impossible, but I do not understand why this is impossible (and more importantly

whether this is the right design pattern in spawn, as it looks to me like certain implementation details from spawn need to be permeated to its clients so that the assumption can be taken without worries)

Make a Comment

Your Name

Your E-Mail

Submit Comment

Follow

Recent Comments

Captain on Express.js Tutorial
Akseli Palen on Make Forever.js Reboot-Proof with Cron
Mayu on Express.js Tutorial
Captain on JavaScript: get the number of properties in an object WITHOUT LOOPING
chris on JavaScript: get the number of properties in an object WITHOUT LOOPING
Armand on Node.js Module – exports vs module.exports
Christian Fei on Make Forever.js Reboot-Proof with Cron
Daniel on How to Write Middleware for Connect / Express.js
Captain on POST / GET Request Handling in Node.js Express
Salman Zaman on POST / GET Request Handling in Node.js Express

Tags

Aptana array CURL database Express
Express.js Firefox git GitHub Google Google
Chrome Gzip Homebrew HTML HTTP HTTPS
JavaScript javascript array javascript for loop
Linux list lol Mac Microsoft MongoDB
Mozilla Mozilla Firefox MySQL mysqldump
node.js NoSQL npm Object pagination
PHP Python Redis Regex Sitemap sitemap.xml.gz
string Thunderbird tuple Ubuntu Wordpress